IPMI Frequently Asked Questions
Last update: 10/30/98

Definitions / Relationships / Diagrams

Q1: What is Platform Management?

A1: Platform Management generally refers to the hardware monitoring, error logging, and recovery features that are built into the firmware and hardware on system platforms. Platform monitoring allows software to check that the server hardware is operating correctly. Key characteristics of Platform Management include the ability to check the inventory, server monitoring, logging of events, and recovery control functions.

Q2: What is Intelligent Platform Management?

A2: The term Intelligent Platform Management refers to autonomous monitoring and recovery features implemented directly in Platform Management hardware and firmware. The key characteristic of Intelligent Platform Management is that inventory, monitoring, logging, and recovery control functions are independent of the system processors, BIOS, and operating system. Platform Management functions can also be made available when the system is in a powered down state.

Intelligent Platform Management capabilities are a key component in providing enterprise-class RASUM (Reliability, Availability, Serviceability, Usability, Manageability) for high-availability systems. Platform status information can be obtained and recovery actions initiated under situations where system management software and normal 'in-band' management mechanisms are unavailable.

Q3: What are IPMI, IPMB, and ICMB?

A3: The *Intelligent Platform Management Interface (*IPMI) is a hardware level interface specification that defines a common, abstracted, message-based interface to platform monitoring and control functions. As a hardware level interface, it sits at the bottom of a typical management software stack. Thus, IPMI is 'management software neutral.' It can be exposed through any standard management software interface, such as WMI, CIM, SNMP, DMI, etc.

The *Intelligent Platform Management Bus* (IPMB) is an I^2C-based serial bus that is routed between major system modules. It is used for communication to and between management controllers. This provides a common interface for

extending platform management hardware and integrating chassis features with the baseboard. The IPMB also provides a common connection for Emergency Management Cards, and other out-of-band access mechanisms, such as the ICMB.

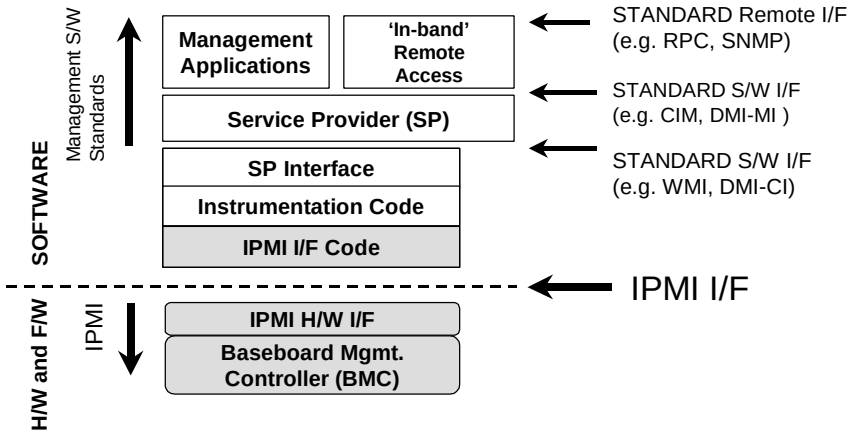
The *Intelligent Chassis Management Bus* (ICMB) provides a common interface for platform management information and control between chassis. The ICMB is designed so it can be implemented with a device that connects to the IPMB. This allows the ICMB to be implemented as an add-on to systems that have an existing IPMB.

Q4: What is the relationship between IPMI and other management interfaces? (WfM, WMI, CIM, ACPI, SMBIOS, DMI, etc)

A4: While IPMI can be used to provide a level of 'stand alone' management, IPMI is best used in conjunction with system management software running under the operating system. This provides an enhanced level of manageability by providing in-band access to the IPMI management information and integrating IPMI with the additional management functions provided by management applications and the OS. System management software and the OS can provide a greater level of sophisticated control, error handling and alerting, than can be solely provided by the platform management subsystem. IPMI resides at the hardware level and provides a standard interface that is complementary to WfM, WMI, CIM, ACPI, SMBIOS, DMI etc.

Q5: What does a software stack with IPMI look like? Where does IPMI fit into the stack?

A5: The following diagram shows the relationship of IPMI to the rest of the management stack:



Q6: What is the relationship of IPMB to SM Bus?

A6: The SM Bus on a system board is typically used to access local baseboard features. Since the IPMB is intended for providing 'public' extension of the platform management features, the SM Bus is normally kept isolated from the IPMB. Depending on implementation, the BMC (baseboard management controller) may share the SM BUS for accessing baseboard information.

Q7: Can SM Bus devices be used on the IPMB?

A7: In general, most SM Bus slave devices may be used on the IPMB since they are both based on I2C. The BMC can be used as a low-level controller to provide system software access to the non-IPMI device. Note however that the IPMB does not support the SMBALERT signal and that some SM Bus devices may operate at 3.3V (or lower) while the IPMB operates on 5V. While SM Bus protocol transactions are allowed on the IPMB, the BMC and other IPMI management controllers can not function as the target of SM Bus protocol operations. That is, IPMI management controllers can not be accessed as SM Bus slaves.

Q8: What is the difference / relationship between spec v0.9 and v1.0?

A8: IPMI v1.0 is a refinement of the v0.9 specification. IPMI v1.0 includes learnings, feedback, and features gathered from industry review and experiences deploying IPMI v0.9 enabled systems. The following presents a brief summary of some of the more significant areas of additions and enhancements to the IPMI v1.0 specification:

- Standardized Watchdog Timer
- Merge of Digital and Discrete sensor classes
- Support for de-assertion event status
- Block Transfer system interface
- Entity Association Records
- Entity Instance field
- Elimination of Device Location code
- Additional Sensor and Event Types
- Improved Identification of Management Controllers
- Sensor Type Assignment for Management Controllers
- Modal Sensor Data Record (SDR) Repository Support
- Simpler Support for Multiple Field Replaceable Unit (FRU) Devices behind a Management Controller

- Simplified Device Locator Records
- Standardized POH Counter
- Improved Baseboard Management Controller (BMC) Messaging
- Power Supply and System Access FRU Information
- Increased commonality among SDRs, sensor commands, and event fields

For more information regarding these areas, see the IPMI V1.0 Specification. The specification is available at http://developer.intel.com/design/servers/ipmi

Q9: Is v1.0 backward compatible with v0.9?

A9: While the majority of commands and interface structures have not changed, there have been enough modifications that v1.0 would not be considered as directly backward compatible to v0.9. Firmware and software based on v0.9 will need some modification and extension for v1.0. It should be noted that v0.9 was an 'early adopter' version of the specification. Backward compatibility with v1.0 will be a major consideration in the specification of future versions of IPMI.

Implementation justification / the future of IPMI / costs

Q10: Why do I want to use IPMI?

A10: IPMI provides several benefits:

- It isolates System Management Software from platform management hardware and low-level server management functions from higher level software.
- IPMI is scalable. An IPMI implementation can be cleanly and quickly extended with new functions, such as additional sensors, management controllers, or FRU inventory devices. In general, these changes are transparent to standard SMS. IPMI allows implementers to "plug in" their IPMI implementation to standard System Management Software (SMS) modules.
- IPMI allows the ICMB (Inter Chassis Management Bus to be implemented as an add-on to systems that have an existing IPMB.
- IPMI supports the WfM (Wired for Management) initiative, which provides a common set of management interfaces to be used by hardware and software vendors. See http://oem.intel.com/ial/wfm/idfwfm.htm for additional information on WfM.

Q11: Where do I find a summary of IPMI benefits?

A11: Refer to the "Background – Architectural Goals" Section of the IPMI V1.0 specification.

Q12: Will IPMI continue to evolve?

A12: Yes, IPMI is a new living industry standard that will continue to improve and grow.

Implementation tools and equipment

Q13: What utilities are needed to implement IPMI?

A13: A DOS-based utility, IPMITOOL.EXE is available for download from the IPMI web site. This utility is provided in binary and source code format. An accompanying README file details command line usage. IPMITOOL.EXE enables issuing of IPMI commands to the BMC and to the IPMB for test purposes.

Q14: What kind/types of lab and debug equipment are necessary?

A14: Recommended tools are dependent on the microcontroller selected for implementation but may include the some of the following:

- An ICE (In-Circuit Emulator) for your microcontroller chip. The availability of a good emulator is a factor in selecting the microcontroller hardware.
- A logic analyzer and an oscilloscope
- I^2C development and test tools.

Implementation requirements

Q15: What is the minimum command set I must use in order to be IPMI compatible or enabled?

A15: Refer to the "Typical Commands for Generic Management Controllers" Section of the IPMI Implementer's Guide for the required and optional commands to implement the IPMI protocol.

Q16: What are the controller / code space / data space requirements?

A16: The requirements are as follow:

- The System Event Log (SEL) typically will require 3 – 8KB of space. The minimum requirement is 16 entries of 16 bytes each.
- The Field Replaceable Unit (FRU) size requirement in the Baseboard Management Controller (BMC) could be up to 256KB. This can vary greatly depending on implementation. Satellite Microcontrollers typically require less FRU space than the BMC.
- The Sensor Data Record (SDR) Repository is dependent on the number of sensors implemented. Compact Sensor Records saves space and require 48 bytes per SDR. Full Sensor Records requires 64 bytes. See the IPMI Implementer's Guide for details.

Q17: Which microcontrollers would you recommend I use?

A17: IPMI does not recommend any particular vendor, type, or family for the microcontroller. In general, a microcontroller with built-in I^2C interfaces is recommended if you're implementing an IPMB. In addition, a controller that has a built-in KCS (keyboard controller style) interface can save board real-estate. "8-bit" class embedded microcontrollers have been shown to provide adequate performance for most IPMI implementations. The following lists some microcontrollers that have been successfully used; Philips* 8xC652 and 8xC552, and Hitachi* H8-300. Other microcontrollers are also available and may be substituted.

Q18: Do I need to use a microcontroller to implement IPMI?

A18: Yes, a microcontroller called the Baseboard Management Controller (BMC) is the heart of the IPMI-based platform management subsystem. If you're integrating a baseboard that already has IPMI, and all you want to do is connect additional FRU (field replaceable unit) information, you will not typically need to add another management controller. If you're extending the baseboard platform management with additional monitoring or control features, you'll typically create a 'satellite' management controller. See the "IPMI System Design Considerations" section of the IPMI Implementer's Guide for details.

Q19: Do I need to implement IPMB / ICMB?

A19: Neither the Intelligent Platform Management Bus (IPMB) nor the Intelligent Chassis Management Bus (ICMB) is required. Implementing the IPMB is highly recommended. Since the IPMB provides a valuable connection for extending baseboard features and a common connection point for Emergency

Management Cards, it is expected that most baseboards that implement IPMI will also provide an IPMB.

Q20: What device addresses can I use on the IPMB?

A20: The addresses that may be used are detailed in the *IPMB Address Allocation* document. This guideline is available from the IPMI web site: http://developer.intel.com/design/servers/ipmi

The *IPMB Address Allocation* document is available under the "Download Supporting Documents" option.

Q21: What is required to add an IPMB peripheral device, such as a power supply or temperature sensor?

A21: The IPMI Implementer's Guide provides the methods and examples for adding various sensors and associated Sensor Data Records.

Q22: What BIOS changes are necessary to implement IPMI?

A22: The level of interaction between the BIOS and IPMI is dependent on the implementation. For example, you can have your BIOS perform actions such as logging POST errors, or use the BMC Watchdog Timer for providing coverage of portions of system start-up. It is also possible to have an IPMI implementation that does not require any BIOS changes, other than providing resource allocation information for the I/O locations used by the System Interface.

Q23: **How much free space should be reserved for adding third-party Sensor Data Records?**

A23: The specification recommends that at least 20% of the Sensor Data Record Repository be left available for platform management extensions.

Q24: How many private management buses are supported by the architecture?

A24: In IPMI terminology, private management busses are those busses that can be accessed via the Master Write-Read I2C command. The command supports up to eight private busses per management controller. You can use other 'hidden' I2C busses that are used by your management for accessing sensor hardware or other board functions. Since these busses would not be accessed using the Master Write-Read I2C command, they wouldn't be considered IPMI private busses. There's no specification limiting the number of these 'non-IPMI' I2C busses in an implementation.

Q25: Will Intel provide source code for a management controller?

A25: IPMI is not tied to any particular type of microcontroller. The internal interfaces to I2C units, interrupt structures, registers, etc., vary widely across microcontroller types. It would be difficult to provide useful source code without favoring a particular controller. Instead, the plan is to provide implementation guides, algorithm information, and tools that can be used for all controllers.

Testing / compatibility

Q26: How do I test commands?

A26: A test utility called IPMITOOL.EXE is presently available from the IPMI web site; http://developer.intel.com/design/servers/ipmi. This DOS-based utility allows developers to issue IPMI commands to the BMC and the IPMB. Check the web site periodically for updates on the availability of other tools.

Support / Resources

Q27: How do I get support?

A27: Send questions to ipmi_qa@ccm.intel.com.

Q28: Where is the conformance suite?

A28: Self tests and plug fest (interoperability) events are planned to facilitate conformance. Details will be provided as they become available.

Q29: Where can I get a copy of the Philips* I^2C specification?

A29: A copy of the Phillips I^2C Specification is available from the following World Wide Web location; http://www-us.semiconductors.philips.com/i2c/

ICMB

Q30: What is the status of ICMB?

A30: The Intelligent Chassis Management Bus Bridge Specification is presently at Version 0.9. The next update is currently under review. Version 1.0 is expected to be released by the end of 1998. The Version 0.9 specification is available from the World Wide Web at http://developer.intel.com/design/servers/ipmi

Q31: How is power supplied to the ICMB in a power-down state?

A31: ICMB controllers and transceivers are required to be on a standby power supply for the managed chassis. This provides access to inventory, discovery, status, and control functions while the chassis is powered down. An always-active tap off the main power subsystem for the chassis typically provides standby power. The ICMB transceivers go into a high-impedance state when they are unpowered. Thus, the ICMB will remain operative if a managed chassis is unplugged from AC.

Q32: Have you considered the potential ICMB ground problem?

A32: ICMB uses RS-485 differential signaling to avoid level problems due to ground potential shifts and noise at points along the cable. Note that the ICMB cable includes a ground wire, so all systems should be on a common AC power ground.

Q33: Will bridges, routers, switches, printers use ICMB?

A33: ICMB can be used for any chassis type. Whether and when a particular type of device uses ICMB is a decision of the vendor. The expectation is that the initial ICMB managed chassis types will include host units, storage expansion units (e.g. RAID chassis), and power and I/O chassis for modular computer systems.

Q34: How do SAF-TE and SES interface to ICMB?

A34: They don't directly interface to ICMB. It is possible to create sensors that provide a parallel, redundant reporting of status and control that is offered by SAF-TE and/or SES.

Q35: Where can I get ICMB cables?

A35: Cables may be ordered from OEI. Call Tino Orialis at 503-844-9000.